Using the Intel® Compilers, 8.0 with the Eclipse Platform Integrated Development Environment on the Linux environment

Table of contents

Introduction

The Eclipse Version 2.1 Platform integrated development environment (IDE) can be used to create C++ applications using the Intel C++ compilers.

This report could be used as a "How-to" guide for using the Intel C++ Compiler in the Eclipse IDE on the Linux platform.

C/C++ development can be made easy in the Eclipse IDE by using the Eclipse C/C++ Development Tools (CDT Version 1.1). CDT provides a set of plugins that implement a C/C++ IDE. It adds a C/C++ Perspective to the Eclipse Workbench that supports C/C++ development with a number of views, wizards, a powerful editor, and a debugger.

This guide assumes that you have correctly installed the eclipse IDE, the CDT plugins, and the Intel C++ compiler on a supported Linux platform. If you have not yet installed the Intel C++ Compiler or the Eclipse C/C++ IDE, please refer to the following getting started guides for additional information. The getting started guide for the Intel C++ compiler is located here. The frequently asked questions for the Eclipse IDE is located here. The getting started guide for the Eclipse CDT is located here.

A Simple Hello World Program

This section shows how to configure the Eclipse IDE to use the Intel C++ Compiler for building a simple HelloWorld Program.
Before starting Eclipse, please make sure that the appropriate environment variables are set for the Eclipse IDE.

Example:

```
export PATH=${PATH}:/usr/local/j2sdk1.4.1_02/bin
export ANT_HOME=~/ant/
export JAVA_HOME=/usr/local/j2sdk1.4.1_02
```

Additionally, environment variables used by the compiler can easily be set with the following script -

. /opt/intel/compiler70/ia32/bin/iccvars.sh

Figure 1 shows a screen capture of the main workbench window that appears on your desktop when you first start your Eclipse IDE with CDT plugins installed -

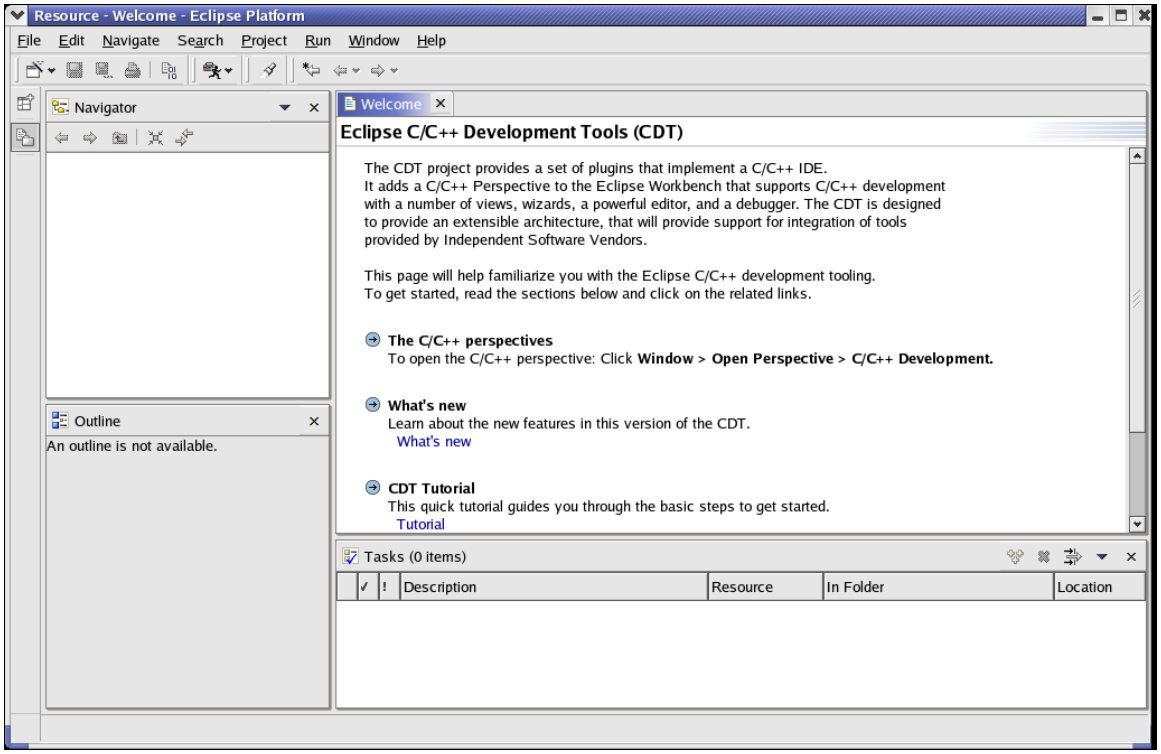

Figure 1

The navigator view (Figure 1, top left) shows the files in the user's workspace, the tasks view (bottom right) shows a list of to-dos,
the outline view (bottom left) shows a content outline of the file being edited and an editor with the CDT welcome page.

To Select C/C++ Perspective, on main menu bar click **Window > Select Perspective > C++ Development**.
The Eclipse C/C++ Development Perspective is as shown below –

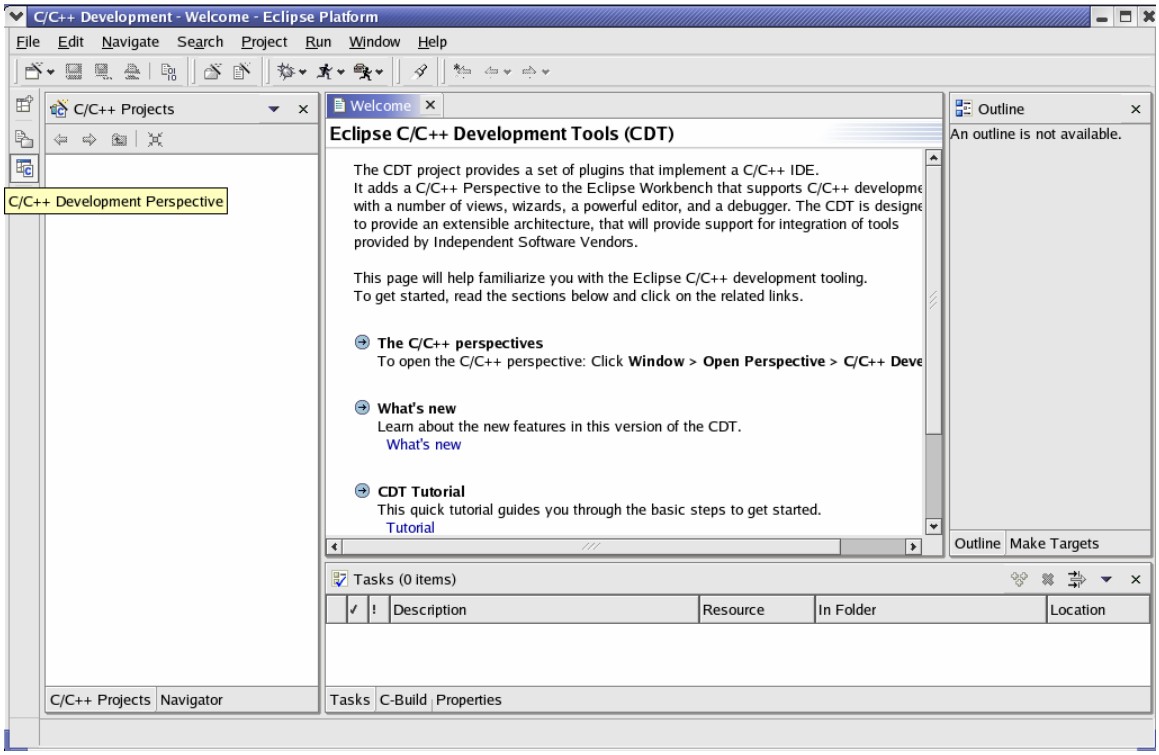

Figure 2

To create a new C++ project, click **File > New > Project,** select **Standard Make C++ Project**, click Next, type **HelloWorld** as project name and then click Finish.

To add files to this project, right click on HelloWorld and select **File > New > File**.
Select C++ File and name it hello.cpp.
Use the Eclipse IDE editor to add content to this file, as shown below -

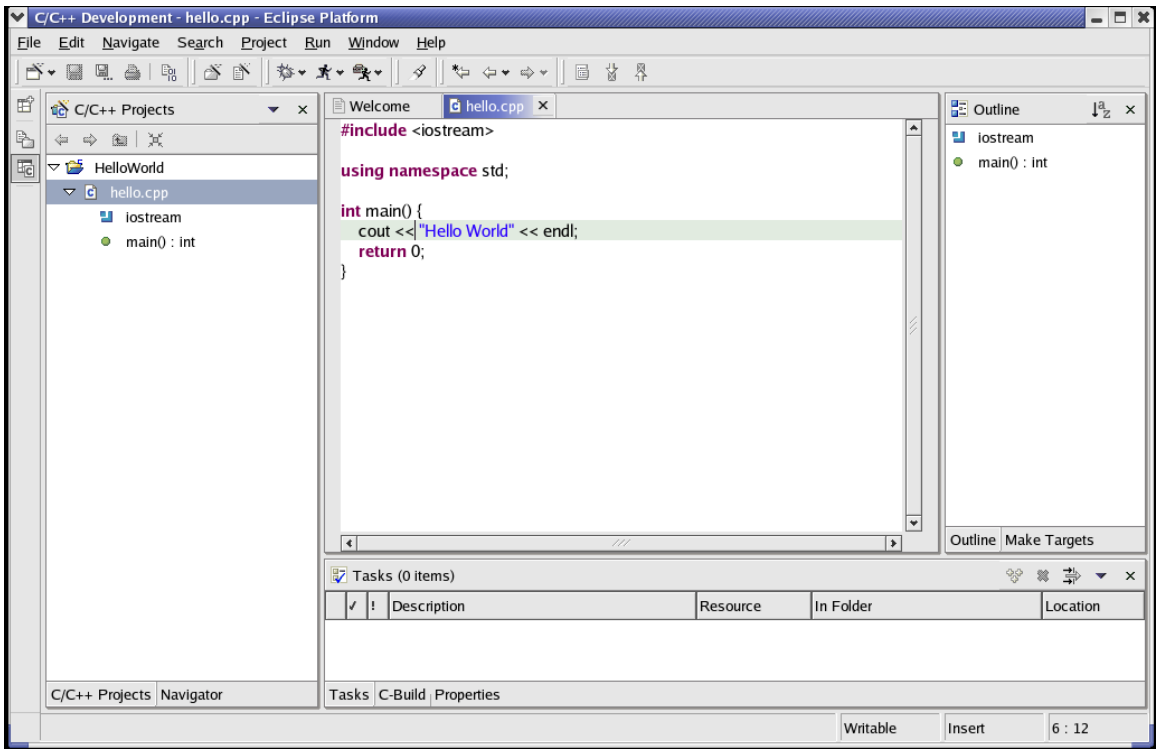

Figure 3

The CDT build process by default executes **make** command to build projects. Hence we need to create a Makefile to specify rules to build the HelloWorld project. The Makefile for the HelloWorld project is as shown below -

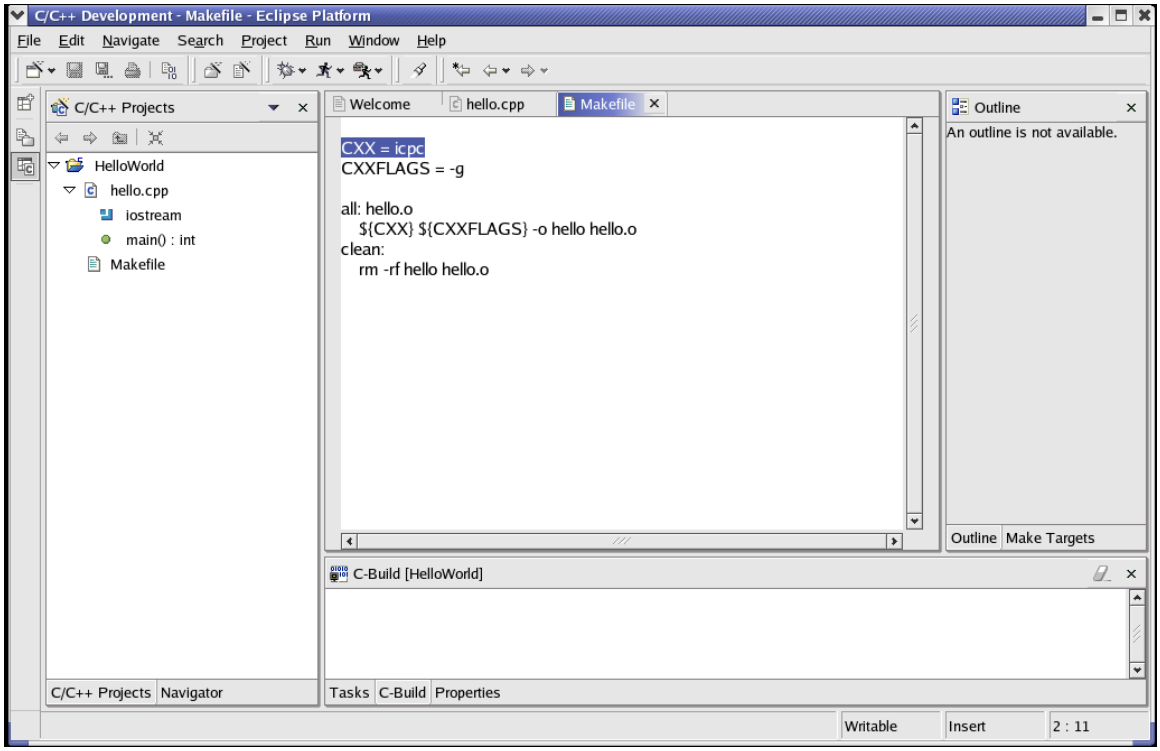

Figure 4

Please note here that the *CXX* variable is set to **icpc** and allows the usage of the Intel C++ Compiler.

To build this project, select **Project > Build Project** from main Menu bar. This event compiles hello.cpp, and an executable gets created in the current workspace. To execute this program, select **Run > Run As > C Local application > hello**

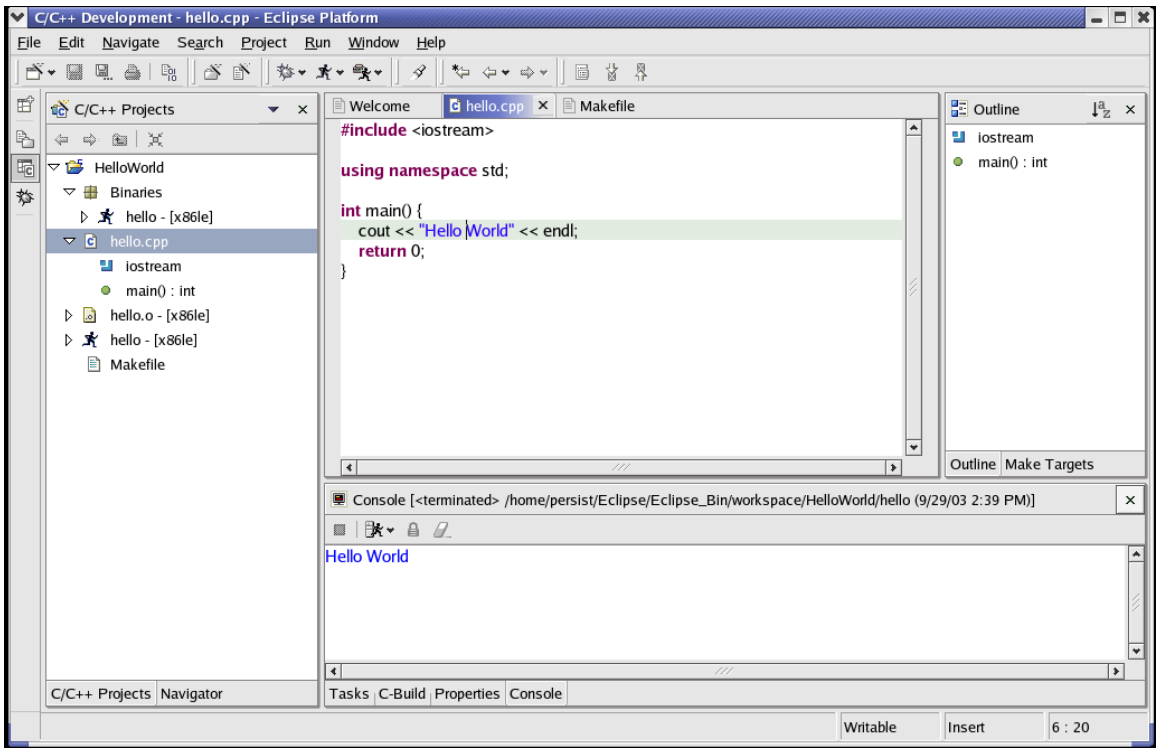

Figure 5

The output of HelloWorld project can be seen in **Console window** as shown above.

Building a more complex application from the Eclipse C++ IDE

We have selected MySQL as a representatative complex open source application. It has a **configure script** to configure the build for a particular platform, and to generate appropriate **Makefiles** using Makefile.in files.

To build MySQL using Intel C++ Compiler through Eclipse, we need to extract MySQL source code in some directory and make the new project point to this location, as described below -

- Create a new project, as explained for HelloWorld project, and name it as **mysql**
- To add files to this project, select **File > Import > File System** as shown below -

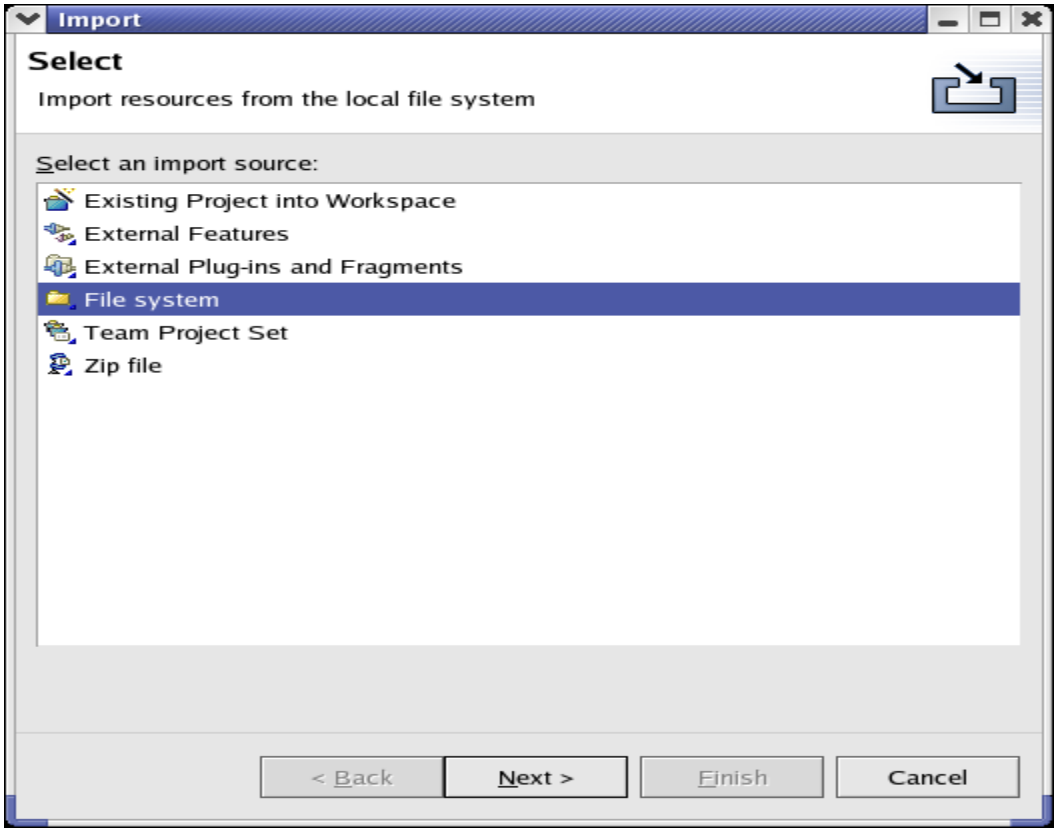

Figure 6

- Browse File system and select MySQL directory. You will see the following directory structure appearing in your C/C++ Projects window -

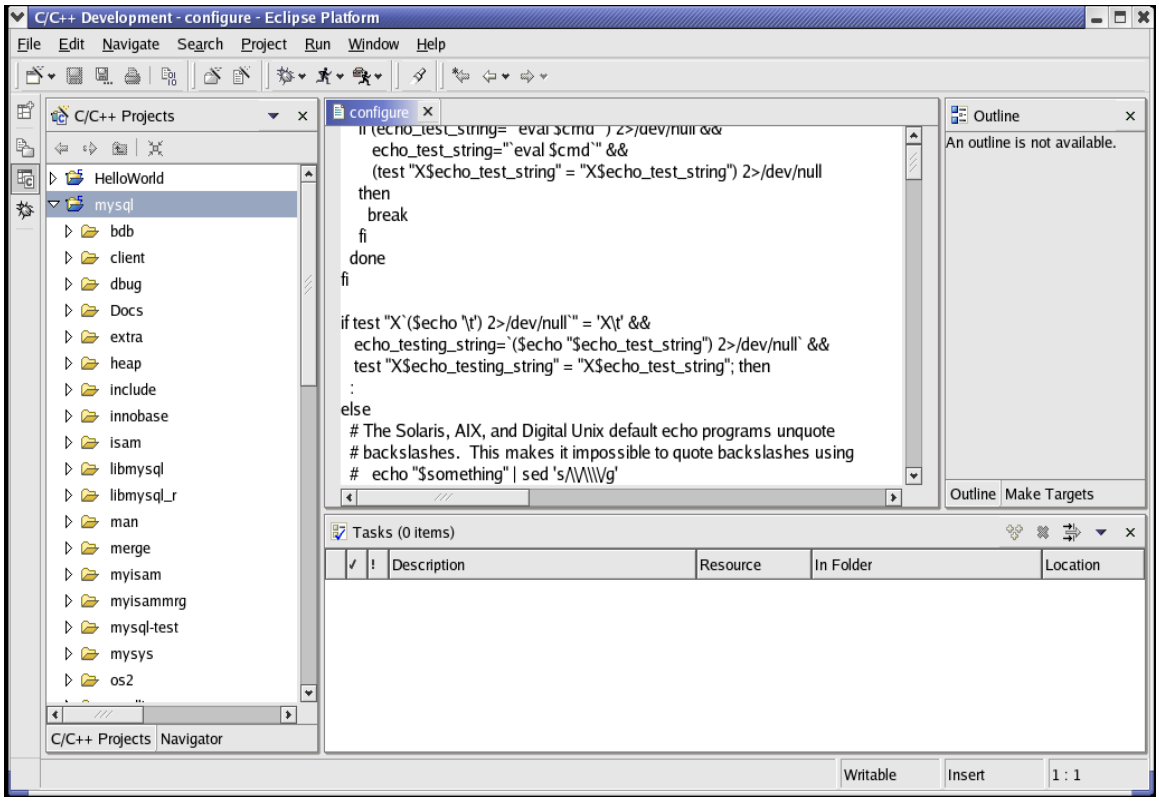

Figure 7

As discussed, Makefile is required to build projects. Makefile for mysql project can be generated using the included configure script as described below -

- Export environment variables to set compiler and command line options
  - export CC=icc CXX=icpc CFLAGS=-O2 CXXFLAGS=-O2
- Execute Configure script to configure it to use Intel C++ Compiler and generate Makefiles
  - ./configure --prefix=~/mysql_install

There is no way to **export variables** through Eclipse IDE; we need to execute this script from the command prompt.
To build mysql, select **Project > Build Project** from main Menu bar.
The generated Makefile and output of make command, in **C-Build window**, is as shown below -

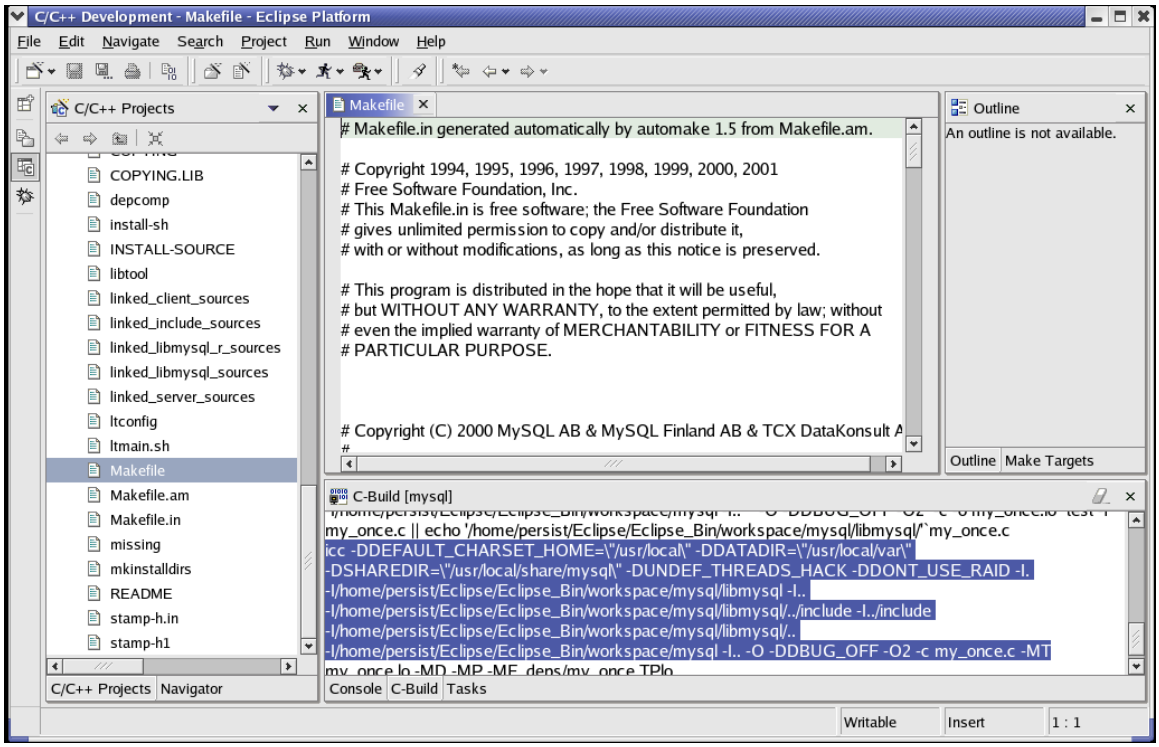

Figure 8

Using Advanced optimization options of the Intel Compilers

The Intel C++ Compiler provides advanced optimization options to improve your application performance on the latest Intel processors.
Some of the optimization options are listed here -

- **-ax{M|K|W|B|P}**: Enables the vectorizer and generates specialized and generic IA-32 code. The generic code is usually slower than the specialized code.
- **-x{M|K|W|B|P}**: Turns on the vectorizer and generates processor-specific specialized code.
- **-ipo**: enables interprocedural optimizations

For more details please refer to the Intel compiler performance optimization guide. Please see Additional Information below for the URL.
The Figure below shows the screen capture of the main workbench window that appears when we build our test project, Vectorization, with -axW option -

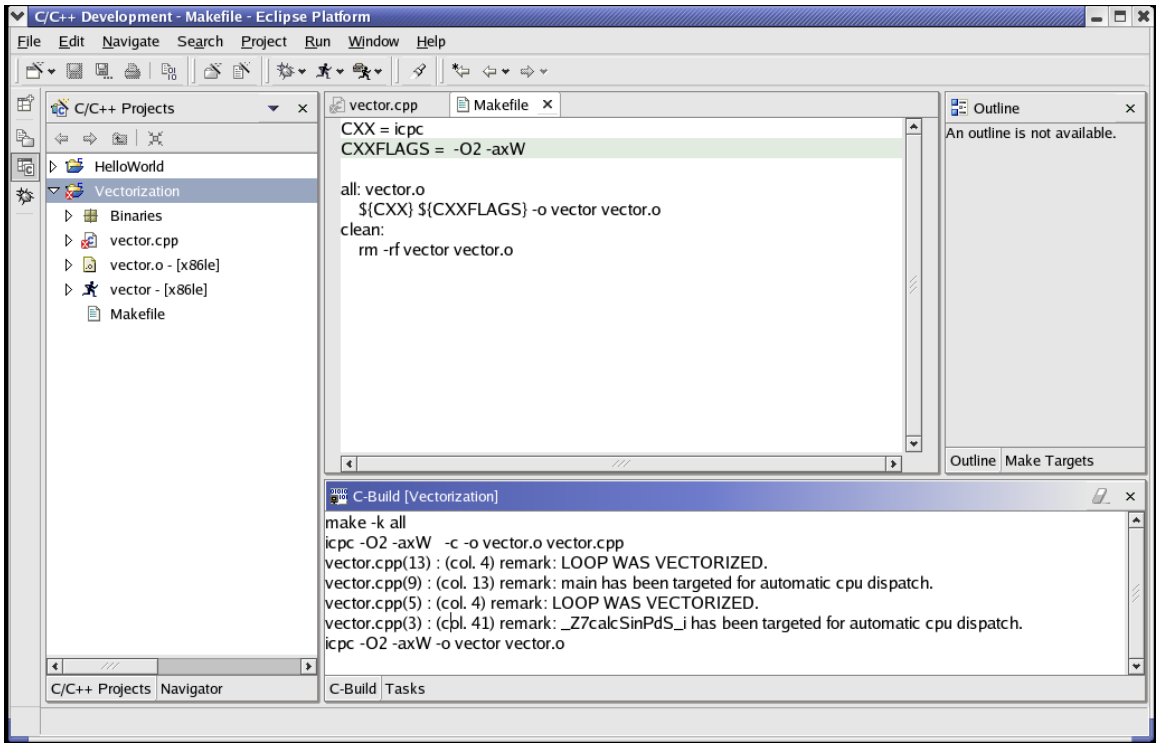

Figure 9

The Intel C++ Compiler vectorizes the loops in vector.cpp and prints the summary of optimization as shown in Figure 9.

Debugging applications

This section explains how to debug an application with gdb through Eclipse IDE.

Here we are assuming that vector.cpp is built with –g option.

To debug this project, select **`Run > Debug As > C local application`** from main Menu bar. Select vector **executable** and gdb as **`a debug configuration to debug`**. This automatically starts **Debug Perspective** as shown below -

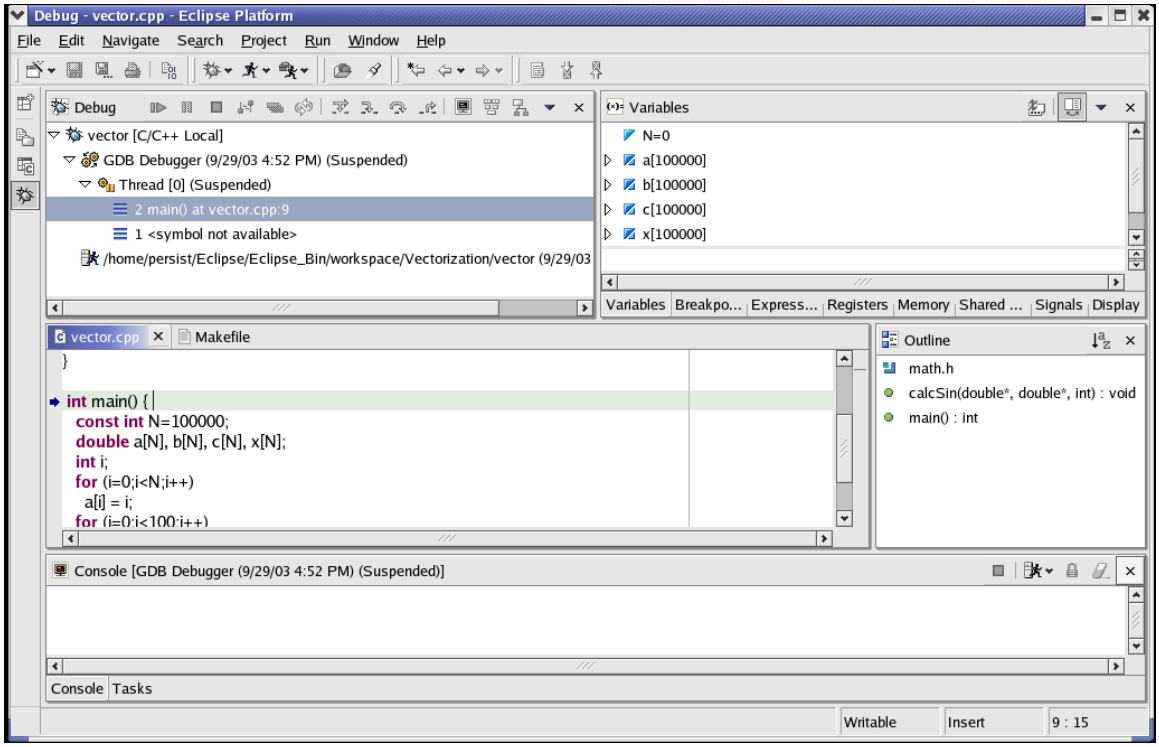

Figure 10

You will find that in Debug Perspective many more options are available under Run option to help in debugging –

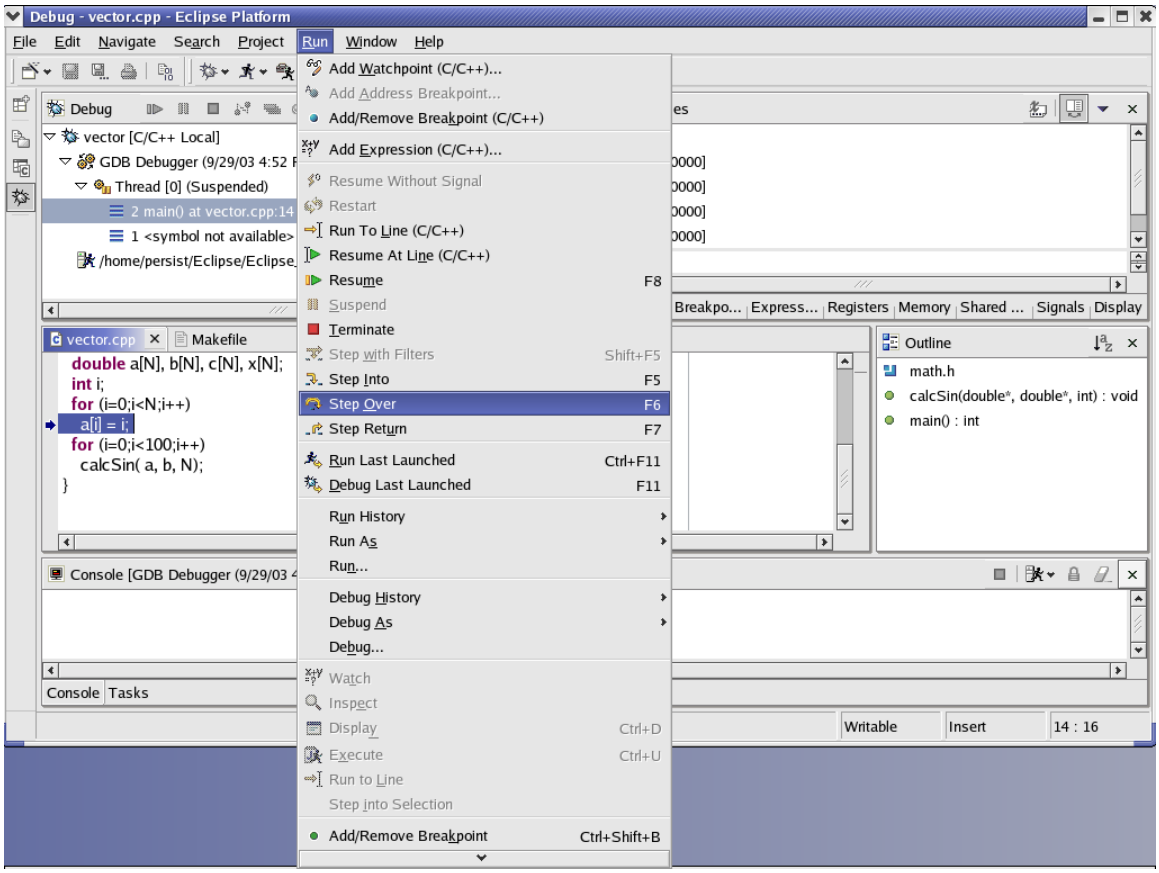

Figure 11

Use appropriate commands to Add/Remove Breakpoints, Add Watchpoints and perform other debug related operations.

Additional Information on the Intel C++ compilers

Support:
http://premier.intel.com

Intel Software products:
http://intel.com/software/products

Intel compilers
http://intel.com/software/products/compilers

Release Notes
Windows:
http://intel.com/software/products/compilers/cwin/relnotes.pdf
Linux:

http://intel.com/software/products/compilers/clin/relnotes.pdf

Getting started Guide
Windows:
http://intel.com/software/products/compilers/techtopics/gettingstarted.pdf
Linux:
http://intel.com/software/products/compilers/techtopics/C_Getting_Started_Guide1.htm

The user's Guide – [Windows] [Linux]
Windows:
http://intel.com/software/products/compilers/cwin/docs/cc_ug.htm
Linux:
http://intel.com/software/products/compilers/clin/docs/cc_ug.htm

Performance Optimization Guide
http://intel.com/software/products/compilers/techtopics/compiler_optimization_71.pdf

Conclusion

There are possibly many ways of integrating your C++ projects with the Eclipse IDE to use the Intel Compilers. In this report we have presented one such approach with the help of Intel C++ Compiler. While this document focuses on the Intel C++ compiler, a similar approach can be used with the Intel Fortran Compiler as well.